instance by replacing "ΣXoff-cent-old" and "ΣYoff-cent-old" with just a certain number of the latest preceding "Xoff-cent" and "Yoff-cent", for instance the previous 99 of each and keeping "n" at 100. This method will lead to consistent representative positions from consistent selected positions quite quickly, but is heavier on memory requirements.

[0053] Another alternative would be to replace formulae (2) and (3) with:

$$Xnew=(Xoff\text{-}cent+[m-1]Xold)/m \quad (2a)$$

$$Ynew=(Yoff\text{-}cent+[m-1]Yold)/m \quad (3a)$$

[0054] where "Xold" and "Yold" are the current X and Y values of the representative positions and "m" is a constant, selected to give sufficient weight to the existing position, so that extreme selected positions are ironed out, for instance "m" may be 100.

[0055] These above approaches rely on calculating an offset from the centre of each key, which means calculating those offsets, in addition to knowing the distance from the selected position to the actual representative position (used in step S106, described above). It is, however, possible to calculate new positions based only on the previous representative position or positions, rather than the centre of a key. For instance, if the old position is considered 99 times more important than the new one, the new representative position would be moved 1/100 of the way from the previous representative position towards the selected position that led to the selection of that confirmed symbol. It is also possible to calculate new representative positions based on averages of the absolute X and Y positions on the screen, rather than relating them to previous representative positions or the centres of the keys.

[0056] Various other possibilities for deciding upon the new calibrated position can easily be used.

[0057] Once the new representative position for a key has been calculated, it is stored in the memory 40 for use in the next run through of the process. Once the representative positions of all relevant keys have been adjusted in step S116, the process reverts to step S100.

[0058] Whilst the above embodiment has re-calibration only for the confirmed symbols, it can operate for every symbol once that is displayed in the message line from a virtual keyboard selection. However, this is more likely to include erroneous selections where the user simply aimed badly and then had to correct.

[0059] A re-calibration system as above without any check on it can be abused, theoretically to the extent that after sufficient use a representative position could bear no relationship to the position of the keys in the virtual keyboard. It is therefore useful to provide a reset function to allow complete resetting of the representative positions. Alternatively or additionally, no representative position may be allowed to wander too far from its original position, for instance in some embodiments outside the display area of the respective key, or in other embodiments farther then halfway towards any of the edges of the key.

EXAMPLE

[0060] An example of the above-described process in selecting a word is now provided. In this example, the user wishes to input the word "this". For this example, the initial letter "t" has already been displayed in the message line, as a first symbol of the symbol string. This was the result of step S108 of the previous run through of the process of FIG. 4. Now the user touches the screen again to put in the letter "h" and touches the screen, at the selected position 52 in FIG. 3. As the preceding input has not yet been confirmed, the previous run through of this process went from step S114 to step S100, without any re-calibration.

[0061] The Sx, Sy values for the selected position 52 are received by the processor in step S100. These are found to correspond to a position in the virtual keyboard in step S102. Thus the user has not selected an item from a list or some other instruction and the previously displayed list can disappear. Candidate keys for the new input need to be determined in step S106, and this involves determining the distances to the representative positions of keys.

[0062] Each of the letter keys is a square of 3 mm by 3 mm, with the stagger between rows leading to a key in one row abutting 0.75 mm of one key in the row below it and 2.25 mm of another key in the row below it. In FIG. 3 the "t" key abuts 0.75 mm of the "f" key and 2.25 mm of the "g" key and the "y" key abuts 0.75 mm of the "g" key and 2.25 mm of the "h" key. In this example, the selected position 52 falls within the display area of the "h" key and is 0.3 mm along from the shared boundary of the "g" and "h" keys and 0.15 mm down from the shared boundary of the "y" and "h" keys. By Pythagoras, the offset distance from the selected position 52 to the representative position of each of the "t", "y", "g" and "h" keys is:

[0063] key t=3.0 mm ($\Rightarrow W_{distance}$=0.33 for the purpose of formula 1)

[0064] key y=1.7 mm ($\Rightarrow W_{distance}$=0.58 for the purpose of formula 1)

[0065] key g=2.3 mm ($\Rightarrow W_{distance}$=0.44 for the purpose of formula 1)

[0066] key h=1.8 mm ($\Rightarrow W_{distance}$=0.55 for the purpose of formula 1)

[0067] Although the distance to the representative position of the "y" key is the smallest offset, as the selected position 52 falls within the display area 54*h* of the "h" key, step S108 still selects and displays the letter "h" in the current position of the message line.

[0068] As at least one candidate is a letter, the next step S202 leads on to step S204. This determines that the symbol currently being input is not the first symbol in the string (as "t" is already there), after which step S206 determines that all the previous symbols in the string have been letter symbols (in this case the only previous symbol was the letter "t"). In step S208 the processor looks at the dictionary database to see if any words are possible. Whilst there are no such words beginning "tt" or "tg", there are some beginning "th" or "ty". Thus the process passes on to step S210, where a set of words is generated for each candidate. The sets generated in this example are:

[0069] For "t"

[0070] "tt" -(W_{freq}=0)

[0071] For "y"

[0072] "type" -(W_{freq}=8)

[0073] "types" -(W_{freq}=8)

[0074] "typed" -(W_{freq}=7)